3. Reading from a File

Write a Ruby function that reads content from a Json file and count the `userId`.

- Function Name : `count_user_ids(path)`

file : file.json

```
┌──(imenⓀhbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ cat 3-main.rb
require_relative '3-read_file'

count_user_ids('file.json')
```

```
┌──(imenⓀhbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ ruby 3-main.rb
1: 10
2: 8
3: 11
4: 13
5: 7
6: 4
7: 9
8: 8
9: 2
10: 1
```

`3-read_file.rb`

```ruby
require 'json'  # Import the JSON module to work with JSON files

def count_user_ids(path)
  # Read and parse the JSON file
  file_content = File.read(path)
  data = JSON.parse(file_content)

  # Initialize a hash to count userIds
  user_id_counts = Hash.new(0)

  # Iterate through the array of objects and count userIds
  data.each do |entry|
```

```
    user_id_counts[entry["userId"]] += 1
  end

  # Sort the results by userId and print them
  user_id_counts.sort.each do |user_id, count|
    puts "#{user_id}: #{count}"
  end
end
```

Example `file.json`

```
[
  {"userId": 1, "title": "Task A"},
  {"userId": 2, "title": "Task B"},
  {"userId": 1, "title": "Task C"},
  {"userId": 3, "title": "Task D"},
  {"userId": 2, "title": "Task E"},
  {"userId": 3, "title": "Task F"},
  {"userId": 3, "title": "Task G"},
  {"userId": 1, "title": "Task H"}
]
```

`3-main.rb`

```
require_relative '3-read_file'

# Call the function with the JSON file path
count_user_ids('file.json')
```

Explanation:

1. **`require 'json'`**:
   - Loads the JSON module to handle JSON file parsing.
2. **`File.read(path)`**:
   - Reads the entire content of the file specified by `path`.
3. **`JSON.parse(file_content)`**:
   - Converts the JSON content into a Ruby array of hashes.
4. **`user_id_counts = Hash.new(0)`**:
   - Creates a hash with a default value of `0` to count occurrences of each `userId`.
5. **`data.each do |entry|`**:

- Iterates through the array, extracting the value of `"userId"` from each entry and incrementing its count in the hash.

6. **`user_id_counts.sort.each`**:
   - Sorts the hash by `userId` (the key) and iterates through it to display the results.
7. **Output**:
   - Prints the `userId` and their respective counts in the format specified.

Expected Output:

If `file.json` contains the example data above, running `ruby 3-main.rb` will output:

```
1: 3
2: 2
3: 3
```

This implementation dynamically handles any number of `userId` values in the JSON file. Let me know if you need further help!